Rktik

A Social Network for Groups

Vincent Ahrend

BACHELOR'S THESIS 2015

PRUEFER
Dr. Helmar Gust, University of Osnabrueck
Dr. Kai-Uwe Kühnberger, University of Osnabrueck

UNIVERSITÄT OSNABRÜCK

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Rktik[1] is an online community that allows like-minded people to collect and publish content together. Rktik is set apart from similar online services by its distinct modelling of personal and collective identity. This thesis describes Rktik's software architecture and how it fulfills its development goals. The first chapter motivates the project and gives an introduction to methodology and comparable products. Then a conceptual description of Rktik's functionality is given. Following that, the technical background of implementation and operation of the service is described. The last chapter contains an evaluation and discussion of the project and its future development.

1.1 MOTIVATION

Rktik is the successor to the Souma app, which started as a hobby project of Vincent Ahrend in 2012 and was then developed from 2013-2014 in the Cognitive Networks study project at University of Osnabrück (CITE). Souma's original motivation was the sharing of files with groups of friends in a service with a modern user interface that does not rely on the benevolence of central service providers. Over the course of development however, the goals of this project changed significantly.

1 The project's name is a made-up word, reminiscent of the German "Arktis" and English "Arctic". It was chosen because it allows for a short domain name (rktik.com). Additionally, the connection to the Arctic, seafaring, and related concepts offer a variety of options for branding Rktik.

At first the aim was to combine social networking functionality with filesharing tools. This functionality can only be provided by a decentral system due to privacy and legal reasons. The *Cognitive Networks* study project developed this idea further and priorities shifted to a focus on the more general goal of a *humanist social platform*. Such a platform would not place economic constraints and existing metaphors for social software systems at the center of its conceptualization, but instead develop solutions from the ideals of humanism.

This shift in the goals of the study project, combined with the remnants of various side projects embedded in the app obscured the core of the project in the code base of Souma. Therefore, after the end of the study project, Rktik was conceptualized as a new web-based platform based on the server-side component of Souma. It has now a much more limited focus: Developing a social publishing platform that implements a new concept of individual and group identity online. Sorted by priority, the goals of this process are:

1. Bringing the application from prototype status to being ready and useful for end users by implementing a complete feature set (**completeness**)
2. Selecting the smallest set of features that allows users an efficient management of privacy and can be implemented by a single developer (**feasibility**)
3. Stripping out and refactoring code from the Souma prototype that is no longer required in Rktik in order to make the project maintainable in the future (**maintainability**)

1.2 PROBLEM: PRIVACY AND IDENTITY

The essential problem in managing privacy in online systems is the need of service providers to directly access user data, which exists for both, cloud-based web applications, as well as native applications[2] that offload parts of their data processing to remote servers (which is the case for many native mobile applications). Companies need direct access to

2 Native applications are run on the user's local machine and stand in contrast to web applications which are run on a server and present their user interface as a web site.

user data, in order to take advantage of central data processing facilities running on web servers. Some of the advantages of web applications, as opposed to native applications without a remote backend, are 1) no software piracy, 2) more efficient maintenance due to direct access to the running system, and 3) no need to support a wide range of (possibly low-performance) user hardware.

One possible solution to the problem of privacy in online systems is presented in the proposed peer-to-peer extension of Rktik (see External Clients). It would approach this problem directly by using cloud infrastructure for transferring data, while processing and storing data on user machines. While this solution does not come with the mentioned advantages of web applications, it offers a lot of value to users who place importance on their privacy. Due to the technical complexity of implementing a peer-to-peer protocol, a second solution to the problem was developed.

The approach pursued in this thesis offers users increased control over their privacy by partially decoupling their offline identity from what they do online. This is achieved by extending the concept of pseudonymity in letting 1) a single person control multiple pseudonyms and 2) letting multiple persons control a shared pseudonym.

1.3 METHODOLOGY

The Rktik application is based on the Souma prototype developed in the Cognitive Networks study project (see 1). In line with the goals outlined in Motivation, Rtkik's development process included

1. Defining required functionality
2. Setting up a development environment
3. Implementing features
4. Setting up third-party infrastructure
5. Finding and removing bugs
6. Evaluation

The Github issue tracker was used to record these tasks and eventual subtasks.[3] A Git repository, also hosted on Github, was used to track the contents of the source code. Changes in the source code are referenced from the issue tracker when they are connected to the resolution of an issue. Using Git and Github helps to keep an overview of tasks and their relation to specific source code changes and offers quick access to past versions of the source code. This feature can be valuable as it allows deleting code without worrying that it might still be needed later on in development.

The Rktik application is implemented in the Python programming language, using the Flask framework for web applications and the SQLAlchemy object relational mapper. Additionally, some features of the user interface are implemented using the jQuery library. The operating environment is provided by Heroku in the form of a platform-as-a-service (PAAS). A PAAS allows uploading and running a web application on provisioned servers without having to configure these servers manually. See Hosting and Deployment for more details on this process.

These choices were taken from the Souma project which used the same technological stack.

1.4 STATE OF THE ART

This section describes approaches to social networks that share similarities with Rktik. I will explain their functionality and specifically focus on their representation of identity and related software features.

This comparison includes three social platforms with more than 100 million monthly active users, each of which has a bias towards a different communication structure. This includes Facebook as a social network used for communication within *peer-groups*, Reddit as an open social network where users publish to the *general public* and email as a communication medium used for *direct communication* with other users. All

3 Access to the issue tracker is restricted to Github accounts registered as Rktik developers.

three services can also be used for one-to-one, one-to-many and many-to-many communication.

In the following, I will explain the basic functionality of each of the services and then compare them to Rktik.

Facebook

Available since: February 2004 Monthly active users: 1.49 billion as of June 30, 2015 ([2]). Website: facebook.com

Facebook is an online social networking service, it allows friends and acquaintances to keep track of each other's personal lifes. Facebook offers a similar core functionality to that of Rktik in that users can publish content on their profiles, vote[4] and comment on content of other users and subscribe to their content feeds. Both services center the user interface on a feed of content from subscribed sources that is sorted based on recency as well as other factors. Facebook also allows users to create groups as either public spaces related to a topic or private spaces for exchange of messages between group members. Users may also send private messages to their "friends".

Facebook is different from Rktik both in policy and features. It requires users to identify with their real name and explicitly forbids creating more than one personal user account. Contents of the Facebook network are hidden from the public by default and accessible only once a friendship connection has been established with a user. Facebook groups can only choose to be either public or private. In contrast to Rktik they can't exchange some messages privately and publish others to the public.

In addition to the core functionality, Facebook offers many more features ranging from business directories to video calling. These have not been included this comparison as they do not directly compare to features of Rktik.

Reddit

Available since: June 2005 Monthly active users: 203 million as of September 15, 2015 (7) Website: reddit.com

4 Votes on Facebook are called "like".

Reddit is a social link aggregator, it lets users create link and text submissions and sorts them based on user votes and recency. Submissions are created in *subreddits*, which are subcommunities of Reddit related to a specific topic. Each submission has a comments section in which its contents are discussed by the community. Comments can also be voted on and are displayed in a hierarchical layout based on their reply-structure. Each subtree is sorted according to number of votes and recency.

Users of reddit can collect *karma*, which is a numeric value that is displayed on their profile and indicates the amount of votes they have received on their own comments and submissions.

This core functionality of Reddit is mostly identical to that of Rktik. Differences between the services can be found in their implementation of user and group identity as well as the way external contents are displayed.

In contrast to Facebook, Reddit encourages the creation of multiple accounts per user (8). However, it does not provide user interface controls for this. Every account is identified by its username, which is pseudonymous and does not necessarily reflect the real identity of a user in any way.

Users can not directly publish content on a user profile or blog, as all submissions must be placed in one of Reddit's communities, which are called *Subreddits*. They allow the aggregation of content related to a specific topic. Their contents are visible either for the general public or only for confirmed members, as decided by the subreddit's founding user. In contrast to Rktik, Reddit does not allow subreddits to aggregate content in a space visible only to members, while also publishing content in a public space.

The content on reddit.com is almost text-only. If a link submission points at an image, a small preview is displayed next to the submission's title, while links to other content are never rendered inline.

Email

Available since: ~ 1971[5] Monthly active users: > 2 billion[6] Website: none

Email is a protocol for distributing messages from an author to one or many recipients. While Rktik, Facebook and Reddit are offered by their respective providers, Email is a *method* of message delivery that is implemented by many service providers worldwide, while still allowing all users to exchange messages with each other. Email messages may also contain any number of media attachments.

A user of Email is identified by their email address. Some email service providers let users choose a username for themselves, while others have policies regarding usage of a real name or other personally identifying features, such as their role in a company. As there is no single user interface for email, users can chose from a wide range of software clients. These can be websites, dedicated email applications, or email features embedded in other applications. Available features depend entirely on the client used.

Email can be used in combination with *automated email lists* for similar use cases as those Rktik was designed for. Automated email lists, or *email lists*, are distribution services that forward incoming messages, sent to a special email address, to all users registered with the email list. Registration may be open to the general public or restricted to a group of users defined by the email list operator. Messages sent to the email list may also be read using a web based discussion archive.

An email list is similar to Rktik movements, subreddits and Facebook groups in that it provides members with a space for discussion and exchange related to a specific topic. Submissions are tied to their author and discussion may be hierarchically organized by reply-structure as is the case in Reddit and Rktik. Email does not provide a mechanism for voting on messages, so discussions are sorted chronologically. Email lists

5 As the email protocol evolved over time from non-networked messaging protocols, there is no specific point in time from which on email was generally available. In 1971 the first networked electronic mail system for ARPANET was standardized (@[Crocker1982]).

6 It is difficult to estimate the number of people who actively use email because of the large number of email service providers. The market research firm Radicati estimated a number of 2.6 billion active email users in 2015 (6).

also do not offer features for collectively authoring content and publishing it to people outside the email list.

In conclusion, Rktik as a social platform has many similarities to existing communication systems. Established concepts and functionality such as user profiles, private messaging, media attachments and others are extended with a novel conceptualization of personal and group identity. These new features extend user's capabilities in shaping their privacy. New group functionality may serve as a catalyst to processes defining the group's online identity.

2 CONCEPTUAL

2.1 OVERVIEW

Rktik is an online community where users share text, links and other media pseudonymously in the context of topic-oriented groups and personal blogs. Contents are sorted based both on recency as well as a voting and subscription system.

This chapter will explain the user-facing functionality of the Rktik website on a conceptual level. First, general concepts are explained, introducing readers to the mechanics of using Rktik. Subsequent sections give detailed information on how Rktik handles *content*, *identity* and *context*.

2.1.1 *Terminology*

This section gives a brief overview of terminology used in Rktik. Please see the relevant subsequent sections for more in-depth descriptions.

Users are individual persons using the site. They may register by creating a *user account* and one or more associated *personas*. Hereby they are able to create content on the site.

(see User Accounts)

All actions of users are attributed to their **active persona**, which is a screen name that identifies them across the site. A user may create any number of personas to shape their privacy, but only one of them can be *active* at a time.

Thoughts are short pieces of text submitted by personas and represent the smallest unit of content.

(see Overview: Thoughts)

Thoughts can link to any number of **percepts**, which are attachments containing either more text or a hyperlink to an external resource. They are displayed alongside the thought throughout the user interface.

(see Attaching Media: Percepts)

Mindsets are collections of thoughts. Any thought that is not a reply to another thought must be contained in a mindset. Every persona has a private and a public mindset (**notebook** and **blog** respectively).

(see [Overview: Mindsets])

Movements are groups related to a specific topic. Each of them has a private mindset for members and a public mindset (blog).

Any persona can create new movements and follow the (public) blog of any movement or other persona.

(see Movements)

2.1.2 *Frontpage*

The Frontpage is located at the root of Rktik and presents users with a stream of thoughts from their movements and followed blogs. It is similar to the Facebook News Feed and the Reddit Frontpage in that the stream is sorted based both on the recency of a submission and the number of votes it has received (see Distributing Attention: Voting and Hotness).

Anonymous users do not have any subscriptions, they are shown thoughts from *top movements*. These are the seven movements with the highest member count.

On the right-hand side of the thought stream, the frontpage also contains a number of other elements:

- The **Frontpage Graph Visualization** displays a visual representation of Frontpage contents as a graph (see Frontpage Graph Visualization).

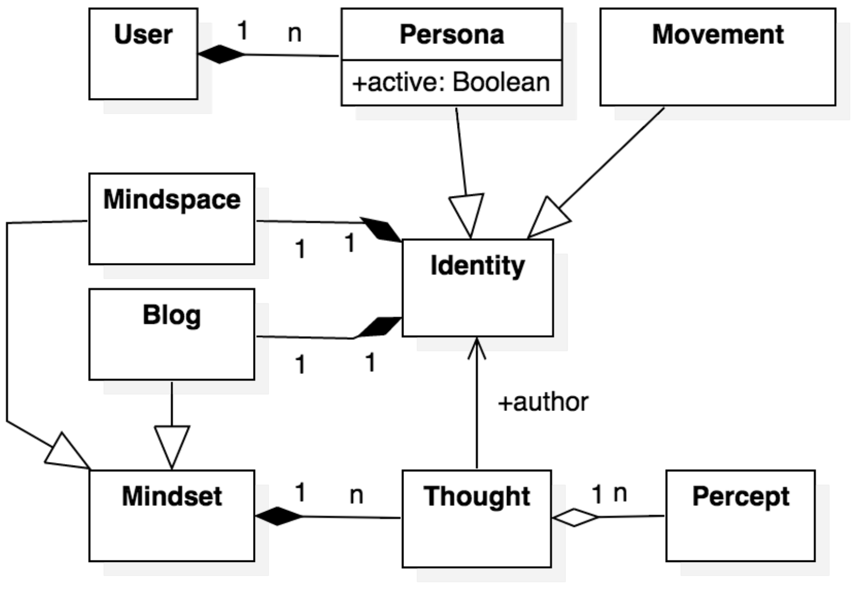


Figure 1: Class diagram of important classes and relations in Rktik as described in Terminology. Most attributes and operations have been omitted for simplicity.

- **Top Thought** When a user is logged in, this contains a short list of thoughts from *top movements* they are not following with their active persona. This allows users to notice particularly popular submissions from contexts they wouldn't see otherwise.
- **Discover Movements** A listing of those top movements the active persona is not following.
- **Recent Thoughts** The most recent publicly visible thoughts submitted throughout the whole site.

2.1.2.1 *Frontpage Graph Visualization*

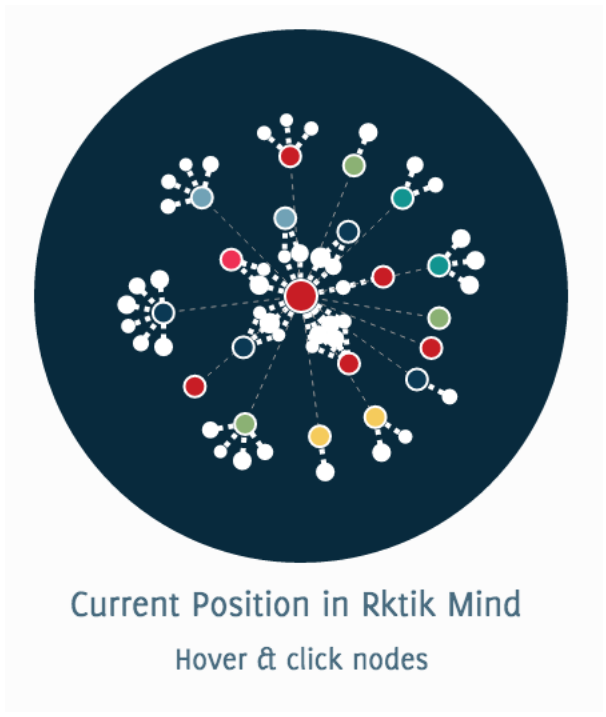


Figure 2: Frontpage Graph Visualization

The Frontpage contains a visual representation of its contents in the form of a graph with a force-directed layout. This layout communicates the flow of information from a persona's subscriptions to their frontpage, while also communicating that not all their content is shown.

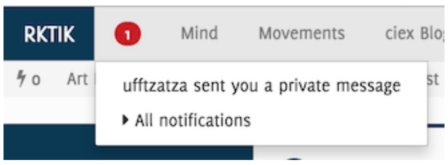
The graph represents the Frontpage as a big red node in the center, identities subscribed to by the active persona (content sources) as medium-sized colored nodes and thoughts as small white nodes.

Thoughts are connected to the center node with a dotted edge if they are currently part of the frontpage stream. Thoughts are also always connected to their author's identity with a further dotted edge. If none of a content source's thoughts are contained in the frontpage, it is connected to the center node with a faint dashed edge. Nodes representing thoughts have a pulsating animation with a frequency correlated to their hotness and capped at 5 Hz.

Thereby, the Frontpage contents surround the center node as a ring of white nodes, pulsating faster or slower according to their position in the stream. They are connected to a second ring of nodes which consists of those movements and personas who submitted the thoughts. Other content sources are located a bit further away to show that their submissions are also eligible for inclusion in the Frontpage.

2.1.3 *Notifications*

Notifications try and catch the user's attention in order to present information which is personally relevant. They are displayed in a drop down menu in the top left corner of every page and some of them are also sent as email notifications.[1]



ufftzatza sent you a private message

▸ All notifications

Figure 3: Notification menu showing one unread notification about a received private message

1 A user may opt out of receiving emails about notifications of a specific type. See User Accounts.

There are four notification types:

- **Reply**: Sent when another user replies to one of the user's thoughts
- **Mention**: Sent when someone uses the @ syntax in a thought to notify a persona directly.
- **Dialogue**: Sent when a new thought is added in a private conversation with another persona
- **Follower**: Sent when a persona's Blog gains new followers

See Notification for implementation details.

2.2 CONTENT

2.2.1 *Overview: Thoughts*

Thoughts are the basic building block for content in Rktik and rougly equivalent to a post on Facebook or submission on Reddit. They consist of a short text, limited to 300 characters, and any number of percepts (attachments). Thoughts are displayed as part of mindsets in different listing styles, 2) on individual thought pages or 3) as part of a chat conversation (see [Overview: Mindsets]).

The restriction on title length has been set for two reasons:

1. A short title reduces the flexibility required from page layouts. Longer titles lead to a bigger variation in title length which would make the usage of separate display styles for long and short titles neccessary and in turn increase development and maintenance time.
2. Short titles require users to be concise when formulating thoughts. In turn, they make it easier for other users to read and understand titles.

Thoughts can be created using the dedicated *create thought* page, which is linked from all mindsets in which the active persona has editing rights, or using the *inline thought creator*, which is embedded in comments sections and as part of the chat widget. The latter only allows posting text content up to the length of a thought's title, but lets users switch to the *create thought* page without losing their input if they wish to continue typing. The dedicated *create thought* page provides separate input fields for title and longform text attachments.

2.2.2 *Reposts*

Thoughts can be *reposted*, which creates a copy of the original thought, but links it to a different mindset and attributes it to the persona who created the repost. Reposts are always created as comments on their original, thereby notifying the original author. Visiting a repost's page, users see the original thought from which it was created displayed in the context area.

2.2.3 *Attaching Media: Percepts*

Thoughts may have any number of attachments for enriching their content. These are either rendered as part of the Rktik website or presented as links to other websites.

Currently the following attachment kinds are supported:

- **Links** Can be attached by embedding a URL inside the thought title or longform text.
  Links to pictures may be rendered inline with the thought. Clicking the picture displays it enlarged in a modal gallery view. If multiple links to pictures are linked to a single thought, the modal view allows browsing through the picture gallery using keyboard and onscreen controls. The display size of pictures is also adapted to the size and number of image attachments (see HTML Templates).
  Links that point to the *soundcloud.com* and *youtube.com* domain will be rendered using the respective embeddable widgets to allow playing music and videos without leaving the Rktik website.
- **Longform text** As thought titles are limited to 300 characters, the longform text attachment allows adding a longer text. This text may be formatted using the Markdown language ([4]), which provides simple markup for basic text formatting such as headlines, enumerations, bold and italic text.

2.2.4 *Distributing Attention: Voting and Hotness*

Personas may vote on thoughts, thereby expressing approval of their content. The number of votes is displayed next to every thought.

Apart from being a visible signal about the number of people who have expressed approval of a thought, votes are also used for sorting thoughts. Depending on context, the order of thoughts is either chronological (chat), reverse chronological (blog) or by their *hotness* value.

Hotness is a numeric value which depends on the age of a thought and the number of votes it has received. It is higher for thoughts more recent and more voted on. Thoughts with equal numbers of votes are effectively sorted in chronological order, while each additional vote pushes the thought upward in the ordering.

Given the number of upvotes v and the number of hours since the thought was created t, a thought's hotness is:

```
hot = v / pow(t + 2, 1.5)
```

This algorithm is adapted from the sorting algorithm used in the social bookmarking site Hacker News (see [9]).

2.3 IDENTITY

Some popular social media sites including Google+ and Facebook have tried to push users into publishing content on their services using a real name (CITE). This was supposed to curb the effect of the online disinhibition effect (CITE) and increase perceived trustworthiness of these platforms (CITE). The practice proved to be controversial, as it did not align with established Internet culture and prevented people from decoupling their online and offline activities (CITE).

However, even though a pseudonymous naming system prevents directly linking offline and online identity based on metadata, such a link can be established by other means. Content posted online may be uniquely bound to an offline identity or link to other services that establish such a binding.

Rktik introduces new tools that allow users to shape their digital privacy. Every user may create multiple online identities and use them to separate areas of their online activity. Users may also join their identities by publishing under the name of a movement. In the following I will explain these concepts in detail.

2.3.1 *User Accounts*

Any user of Rktik can register a personal user account which allows them to create content and vote on submissions. Creating an account requires a valid email address, a password and a name and color value for the user's first persona.[2]

The account feature serves two main purposes:

1. Authorizing a user's identity when they start a session of using the site, by asking them to enter the account-specific password and email-address combination. Authorized users may act on the site as one of the personas connected to the user account.
2. Obtaining a valid email address, so that users may be sent notifications and other messages related to their activity on the site.

User accounts also store the user's email preferences allowing users to disable emails of specific kind (see Notifications). In general, emails should be sent as little as possible as users may perceive them as spam if they carry insufficent personal relevance or value.

2.3.2 *Personas*

The personal identity of a user of the Rktik website is partially decoupled from their physical identity: While users may choose to use their real name, they can also use one or more pseudonyms.

Other online communities allow this to different extents: On Reddit, a user may choose an arbitrary name for their user profile, while Facebook asks their users to provide their real name for online communication. Rktik goes one step further by not only allowing arbitrary handle

2 The color value is used to decorate a persona's username in the site's design.

names but also featuring arbitrary *numbers* of handles, which are called *personas*.

On initial signup, new users create their first persona by giving it a name and assigning a color to mark this persona's submissions. Following completed signup, users can create more personas using the *switch* menu in the upper right corner of every screen. They can also to choose to create a new persona any time they are joining a movement. Therefore, they can keep their membership in that movement separate from activities in other parts of the site.

Other users can not tell whether any two personas are linked to the same user account.

2.3.3 *Personal vs. Group Identity*

Groups on other websites have an identity communicated by the contents of the group and other markers related to the group's representation on the website (group name, picture, color scheme). Rktik furthers this concept by communicating not just identity but also a movement's *agency* in 1) language as well as 2) features that suggest group agency.

A movement's space for internal discussion and exchange is called *mindspace*, implying that the movement has a shared mind, the contents of which are displayed in this place. This notion is also reflected in how users place contents in a group: They *create thoughts* in the *mindspace*.

A movement's agency is further implied in the functionality of the movement blog. Its contents are not dictated by a designated member of the movement, but selected by personas voting on thoughts in the movement. The movement members collectively put into action decisions, which are then attributed to the movement as a whole, when they are displayed on the site. Please see section Movement Agency for possibilities for further development of the concept of movement agency.

—

On the one hand, users have the option of not just operating under one identity but assuming any number of Personas while using the site. These Personas may reflect different social roles, they can be used to voice unpopular opinions or generally content a user doesn't want to identify with their real name.

On the other hand, movements, which reflect groups of users, share many features with individual users. They have a blog which allows them to publish contents in the name of the movement and have a place for internal dialogue, similar to how an individual user can use their notebook to remember content.

2.3.4 *Movements*

Movements allow groups of users with a shared interest to exchange their thoughts about it. Any user may create new movements by specifying a name and optional mission statement. A movement may also be created with the *private* option, which will hide contents of the movement mindspace from non-members and only allow users with an invitation code to join the movement as a member. Invitation codes may be created by any movement member.

Each movement has its own blog and mindspace with embedded chat room. Members can post thoughts to the movement mindspace by using the *Create Thought* interface or posting to the movement chat. Mindspace contents are sorted by their hotness value (see Distributing Attention: Voting and Hotness).

2.3.4.1 *Promoting Content*

Alongside each thought in the mindspace a progress bar is displayed which indicates how many more votes are required for a thought in the mindspace to be promoted to the Movement blog. This threshold value depends on the number of members of the group. Given the number of members c it is defined as:

```
threshold = round(c / 100 + 0.8 / c + log(c, 1.65)) if c > 0 else 1
```

[TODO: picture showing relation threshold and number users]

This formula ensures that a low number of votes is required while a movement is small, creating lots of content on the blog, while a large movement requires more votes relative to its user count so that only the best content will be posted to the blog.

Movements are similar to Facebook Groups, Reddit's Subreddit feature and Email newsgroups. All of them distinguish between private and public groups as Rktik does, however they don't provide group members with the ability of democratically deciding on content to be published separately from the group pool to a public medium.

2.3.4.2 *Private Movements*

When creating a new movement, a user may chose to make it *private*, which 1) hides the movement mindspace from non-members and 2) only allows users with a valid invitation code to enter the movement.

Invitation codes can be sent by movement members by either entering the email addresses of invitees or by copying a URL with embedded invitation code and sending this to the invitee (e.g. using a messenger application).

The blog page of private movements indicates the movement founder so that users interested in joining may contact this founder and ask for an invitation.

2.4 CONTEXT

Every thought is linked to the context in which it was created. This can be another thought when a user is submitting a reply, it can be the mindset in which they create the thought or it can be both of them.[3] Mindsets are collections of thoughts, owned by identities. There are three different kinds of mindsets for 1) internal thoughts of an identity (mindspace), 2) its published thoughts (blog), and 3) private conversation (dialogue). Each of them is rendered with a particular layout and functionality.

This section will show how communication happens in the context of a single thought, as well as in mindsets. Then, the differences between

3 If a thought is defined as a reply to another thought and also defines a mindset as context it is included in both contexts. Its rendering in each of the contexts also refers to the other side, effectively linking discussion in two separate areas of the website.

the three kinds of mindsets are explained though their requirements, tasks and conceptual design.

2.4.1 *Threaded Discussion*

Every thought in Rktik has its own page, which collects all information related to the thought. This includes both its text content and percepts as well as the thought's context, metadata and reactions to it. Reactions may be replies written by other users, reposts and automatic promotions (see Promoting Content).

Displaying reactions to a thought in a flat listing can make it harder for readers to follow the exchange, as conversations regarding different aspects of the original thought may be interweaved in the listing. Rktik solves this problem by using a hierarchical display of reactions. Direct replies are aligned to the left-hand side of the screen with subsequent replies indented to the right. Every subtree of the discussion is sorted by hotness.

The reaction tree depth is limited to three levels for performance reasons. If a reaction is located in a mindset different from that of the original thought, reactions happening in the other context are also included up to a total depth of three levels.

If the original thought was created as a reply itself, the page also contains its parent thoughts. The thought's author may define from 0-10 levels, how deep the reply-chain is recursed upwards for this purpose by setting the context-depth setting on the left side of a thought's page. Setting it to zero hides the thought's context from other users and removes it from the discussion pages of these ancestors.

2.4.2 *Contextual Rights Management*

Depending on context, a different set of identities is authorized to create, edit and delete thoughts in a given mindset. See [Rights Management] for detailed information about which users are authorized for which actions.

2.4.3 *Mindspaces*

Mindspaces are the first of three kinds of mindsets. They collect internal thoughts of an identity, as opposed to thoughts published to all users of Rktik. Both personas as well as movements are identities in Rktik and therefore have their own mindspace.

Persona Mindspace

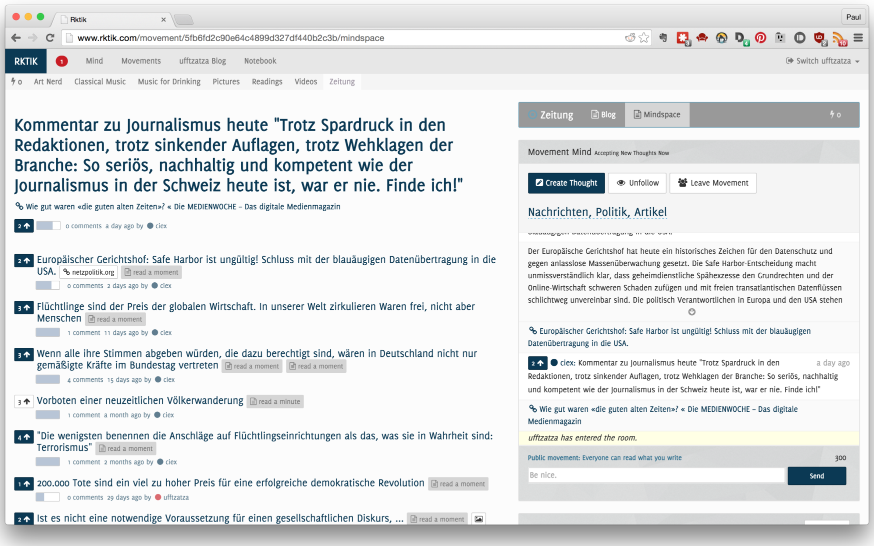
Every persona has access to their private mindspace in which only they can read and write. This makes it a space for collecting thoughts before deciding on whether or not to to publish them, or in which context to publish them. The mindspace of a persona is also refered to as the persona's *notebook* to make it easier for new users to understand how they can use this feature.

Apart from creating thoughts directly in this mindset, users may also use the *repost* interface (see Reposts) to copy thoughts from anywhere on the site into their notebook.

Movement Mindspace

The movement mindspace is the primary interface for movement members. This space is facilitating discussion and exchange between movement members and serving as a staging area for content that might be posted to the movement's blog. The movement mindspace should therefore 1) provide an overview of the most interesting content recently posted to the mindspace and 2) provide members with the sense that they can communicate directly and immediately with each other. To fulfill both of these requirements, the layout displays thoughts both as a listing sorted by hotness as well as using a chat widget.

While the chat is an automatically updating view of the most recent thoughts, the listing changes slower. As the hot ranking sorts based on recency and number of votes, it is akin to a rolling toplist of the currently most interesting thoughts. These thoughts can collect more votes in the listing view until they reach the threshold for pomotion to the movement's blog (see Promoting Content). A short blue bar displayed underneath each listing entry indicates how many further votes are required for a promotion.



Kommentar zu Journalismus heute "Trotz Spardruck in den Redaktionen, trotz sinkender Auflagen, trotz Wehklagen der Branche: So seriös, nachhaltig und kompetent wie der Journalismus in der Schweiz heute ist, war er nie. Finde ich!"

Wie gut waren «die guten alten Zeiten»? « Die MEDIENWOCHE – Das digitale Medienmagazin

2 0 comments a day ago by ciex

2 Europäischer Gerichtshof: Safe Harbor ist ungültig! Schluss mit der blauäugigen Datenübertragung in die USA. netzpolitik.org read a moment
0 comments 2 days ago by ciex

3 Flüchtlinge sind der Preis der globalen Wirtschaft. In unserer Welt zirkulieren Waren frei, nicht aber Menschen read a moment
1 comment 11 days ago by ciex

3 Wenn alle ihre Stimmen abgeben würden, die dazu berechtigt sind, wären in Deutschland nicht nur gemäßigte Kräfte im Bundestag vertreten read a moment read a moment
4 comments 15 days ago by ciex

3 Vorboten einer neuzeitlichen Völkerwanderung read a minute
1 comment a month ago by ciex

4 "Die wenigsten benennen die Anschläge auf Flüchtlingseinrichtungen als das, was sie in Wahrheit sind: Terrorismus" read a moment
1 comment 2 months ago by ciex

1 200.000 Tote sind ein viel zu hoher Preis für eine erfolgreiche demokratische Revolution read a moment
0 comments 29 days ago by ufftzatza

2 Ist es nicht eine notwendige Voraussetzung für einen gesellschaftlichen Diskurs, ... read a moment

Movement Mind Accepting New Thoughts Now

Nachrichten, Politik, Artikel

Der Europäische Gerichtshof hat heute ein historisches Zeichen für den Datenschutz und gegen anlasslose Massenüberwachung gesetzt. Die Safe Harbor-Entscheidung macht unmissverständlich klar, dass geheimdienstliche Spähexzesse den Grundrechten und der Online-Wirtschaft schweren Schaden zufügen und mit freien transatlantischen Datenflüssen schlichtweg unvereinbar sind. Die politisch Verantwortlichen in Europa und den USA stehen

Europäischer Gerichtshof: Safe Harbor ist ungültig! Schluss mit der blauäugigen Datenübertragung in die USA.

2 ciex: Kommentar zu Journalismus heute "Trotz Spardruck in den Redaktionen, trotz sinkender Auflagen, trotz Wehklagen der Branche: So seriös, nachhaltig und kompetent wie der Journalismus in der Schweiz heute ist, war er nie. Finde ich!" a day ago

Wie gut waren «die guten alten Zeiten»? « Die MEDIENWOCHE – Das digitale Medienmagazin

ufftzatza has entered the room.

Public movement: Everyone can read what you write 300

Be nice.

Figure 4: Movement mindspace showing thoughts sorted by hot ranking on the left and movement chat on the right

2.4.3.1 *Chat*

The chat should provide members with a sense of direct and immediate participation in the movement. It provides an automatically updating, chronological listing of all events related to the movement, including all thoughts submitted to the movement mindspace and thumbnails of media attachments.

This allows for a different mode of communication from the listing view and threaded discussion. The immediate transmission of messages allows the use of language with a conversational tone and creates the sense of a more direct exchange between participants.

Replies to any thought in the movement mindspace are also displayed in the chat, with an annotation that explicitly marks them as replies and provides a hyperlink to their context. This allows movement members watching the chat to directly start participating in those discussions.

Below the chat message listing, an inline form UI allows users to send thoughts to the mindspace/chat. Submitting the form clears the input without reloading the page, so that the user can enter another message immediately. Above the input, a notification text informs the user about the privacy of messages entered, which is dependent on whether the corresponding movement is private or not (see Private Movements).

2.4.4 *Blogs*

Blogs are mindsets which allow identities to share their thoughts with a wider audience and are sorted in reverse chronological order. Any persona can follow any blog. Doing so places the blog's contents in the pool of thoughts eligible for that user's personal frontpage.

Personas can directly post new thoughts to their personal blog, while movement members can only indirectly place content in a movement's blog through voting (see Promoting Content).

2.4.5 *Dialogue*

While mindspaces allow exchange between many users, and blogs allow broadcasting to many users, the dialogue mindset models a conversation between just two participants. As it is also implemented as a mindset, messages can be reposted freely between a dialogue and any other context.

Apart from the different privacy setting, a dialogue provides the same affordances as the chat module in a movement mindspace (see Chat).

3 IMPLEMENTATION AND OPERATION

3.1 OVERVIEW

This chapter gives technical information about the implementation and operation of the Rktik service.

It is divided into sections for the shared data model *Nucleus*, the *Glia* web server, about improving performance as well as the deployment and operation of Rktik in a hosted environment.

3.2 SHARED DATA MODEL: NUCLEUS

The Nucleus library uses the SQLAlchemy ORM[1] to provide data persistency and defines methods for context-independent data processing. It is implemented as a Python package which can be imported from the main application *Glia*. The Nucleus package also provides a direct database connection that can be used from Glia to bypass the ORM layer as well as a connection to the in-memory cache *memcache*, which is also extensively used by the ORM models. A signalling namespace provides event hooks which are used to automate postprocessing and other actions in reaction to model changes.

As Rktik was planned as a semi-decentralized service[2], the object relational manager was decoupled from the rest of the application from

1 SQLAlchemy 0.9.2

2 Semi-decentral in this case means that users can chose between 1) a client application for rendering and processing data which uses a web server for the transfer of (encrypted)

the beginning to allow for the easy implementation of client and server applications using this codebase.

3.2.1 *Serializable*

The *Serializable* module primarily provides serialization capabilities to ORM models that inherit from it. This functionality is not in the scope of this thesis, but part of the planned P2P extension (see External Clients). As the module is also required for rights management, I have left it in the codebase submitted along this document, and will describe the relevant functionality here.

Serializable objects provide a method *authorize*, which validates that a given user may execute a specific action on the instance. Each model that inherits from Serializable can define its own handling of user rights by overriding this method. Please see [Rights Management] for detailed information on which users are allowed to make changes to which objects.

3.2.2 *Nucleus Models*

The module `nucleus.models` contains defitions for all ORM models. Each model is represented by a class definition. See [API specification] for technical details such as the attributes and methods provided by each model. This section gives an overview for each model's properties and responsibilities with respect to the functionality described in the *conceptual* chapter.

3.2.2.1 *User*

The `User` model represents a registered user of the site. It has relations to all personas of this user and stores basic metadata such as the user id, account creation data, email and password hash. The `User` class is also

data and 2) solely using Rktik on its website without installing an application on their computers.

used for verifying email validation actions and storing the validation state related to the user.

3.2.2.2 *Identity*

The `Identity` class is a superclass for `Persona` and `Movement`as these two share many attributes and methods related to them being identities.

Apart from basic information such as username, user color, creation and modification timestamps, the `Identity` model has relations to the blog and mindspace associated with each instance.

3.2.2.3 *Persona*

The `Persona` class represents personal identities taken by users of the site. Each `User` instance may be connected to many `Persona` instances.

The `Persona` model provides methods for toggling instances' membership in movements and following and unfollowing blogs. It also provides a number of cached methods that serve as shortcuts to information related to the Persona which is computationally expensive to collect (see Improving Performance).

3.2.2.4 *Movement*

Just as the `Persona` model, `Movement` instances inherit from the `Identity` model and thereby provide all its attributes and methods. They also store the movement's mission, whether the movement is private and relations to the movement's admin (founder) and to movement members.

MovementMemberAssociation

The members relation is implemented using the association object pattern to store additional metadata about the membership:

- Membership active status. Inactive memberships are used to represent Personas who have left the movement and for invitations, which are created as inactive memberships with no associated persona.
- Timestamps for creation and last modification of the membership.

- The member's role in the movement (currently one of "member" and "admin").
- Time when the persona was last present in the movement chat

3.2.2.5 *Mindset*

This model represents a *set of thoughts* with an author and is a superclass of `Mindspace`, `Blog` and `Dialogue`.

- **Mindspace** models internal thoughts of an identity
- **Blog** models a blog publication
- **Dialogue** models a conversation between two identities. The dialogue model has an additional relation to personas representing the "other" of a conversation. This means that retrieving the dialogue between two given personas is not a simple lookup, as the `author` and `other` attribute can be filled interchangeably. Therefore, a `get_chat` classmethod is provided, that tries the two lookup possibilities in succession and returns a new dialogue instance if both are unsuccessful.

3.2.2.6 *Thought*

The `Thought` model represents content submissions by users of the site. Each instance stores the title text and metadata of the thought. All other media related to the thought is contained in `Percept` objects. Thoughts also store the context they were posted in, which may either be a parent thought for replies or a mindset for top-level thoughts.

The thought class is able to generate instances of itself directly from text input received via the UI through a classmethod. This process includes detecting embedded URLs and validating whether they refer to a valid HTTP resource, creating `Percept` objects for any text, link or linked picture attachments, relaying notifications triggered by the creation of new thought and percept instances and invalidating caches touched by the new thought.

Thoughts also have a relation to their votes and helper methods for accessing information about these votes (has a specific user voted, total amount of votes, hotness value).

3.2.2.7 *Upvote*

The `Upvote` model inherits from `Thought`. Its instances represent votes cast by personas on other thoughts. This relation is represented by the upvote instance referring to the voted thought as its parent.

3.2.2.8 *Percept*

The `Percept` model represents attachments on thoughts. The `Percept` class is used as an abstract class with subclasses:

- `LinkPercept`, `LinkedPicturePercept`: Store a URL link, which is rendered inline in case of the `LinkedPicturePercept`
- `TextPercept`: Stores the attached text
- `MentionPercept`: Stores a relation to the linked user and the text used to refer to them (these might be different if the mentioned persona changes their username after being mentioned)
- `TagPercept`: Store a relation to an instance of the `Tag` model.

Percepts are linked to a thought with the association object pattern. The `PerceptAssociation` class stores the author who created the association in addition to its thought and percept. The association's author is usually identical with the thought author, but movement admins also have the rights to edit thoughts submitted to their movement's mindspace.

3.2.2.9 *Tag*

The `Tag` model represents a label attached to a thought by embedding words starting with the hash character '#' in the thought title or text. Tagged thoughts don't have a direct relation with a tag instance but use the `TagPercept` model as an association object. This way, tags can be renamed globally without touching all thoughts that have this tag.

3.2.2.10 *Notification*

Notifications represent direct messages to the user, generated automatically when certain events require the user's attention. The `Notification` base class stores metadata such as the notification text, URL, unread status and recipient. Subclasses are used to represent specific kinds of notifications:

- `MentionNotification`: Sent to a persona when a mention referring them is posted
- `ReplyNotification`: Sent to a persona when they receive a reply on one of their Thoughts.
- `DialogueNotification`: Sent for new messages in a private conversation between two Personas.
- `FollowerNotification`: Sent to a persona when their blog gains a new follower.

3.2.3 *Modeling Data with SQLAlchemy*

SQLAlchemy allows the implicit specification of database schemas through defining the Python classes the database ought to model. It maps user-defined Python classes to database tables and instances of these classes to rows in the tables. Changes to instances are transparently synchronized with database contents and queries for retrieving data can be formulated in an object oriented expression language. This has the advantages that 1) developers can start modifying the database schema without having to learn a query language specific to the used database, 2) the connected database backend can be changed with minimal modifications to the model specifications and 3) all code related to the ORM models resides in one place, limiting code fragmentation.

While SQLAlchemy makes getting started really easy, it can also lead to performance problems. Reducing the complexity of database access is appropriate for straightforward use cases but can lead to inefficiencies in more complex scenarios. Many advanced queries can be optimized with some knowledge of how the underlying database is used, as SQLAlchemy does not necessarily translate a given command into the most effective query. The library provides an extensive suite of tools for implementing these optimizations.

When model definitions are changed, while the database is already used in production, it is not enough to recreate the database using the new schema, as old data may have to be migrated. Rktik uses the Alembic library[3] to record schema changes and migrate the database layout. Schema migrations are automatically executed on the server by the deployment script (see Hosting and Deployment).

In cases where not only the database schema, but also its contents have to be modified, migration scripts have to be manually written in accordances with the changes. These are stored in the `glia/migrations_extra` directory for one-time execution on the server.

3.3 WEB SERVER: GLIA

The *Glia* web server is responsible for collecting and computing contents of the user interface, serving asynchronous UI updates, validating, storing and modifying information entered by the user, automatically performing maintenance operations and scheduling email delivery.

The Glia web server consists of these components:

- *Views* are functions mapped to URL patterns and compute their contents when accessed by a user
- *Websocket events* are special views used for asynchronous communication with a web browser
- *Forms* validate restraints on structured user input
- *HTML Templates* are used to map data into a graphical layout to be rendered by a browser
- *Configuration files*
- *Database migration scripts*
- *Static files* (Images, frontend Javascript resources, etc)

Session Management

Session management is responsible for storing information about which user is logged in on which browsers. Rktik uses the Flask-Login extension[4] to provide most of this functionality.

3 Alembic 0.7.5.post.2
4 Flask-Login 0.2.11

Users can login using their email and password which, given a correct password, will let Flask-Login store a cookie in their browser recording the logged-in state.

3.3.1 *Web View and URL Routing*

Views are functions that return HTML content and are mapped by a route to a URL scheme, which can be acessed by a user through a web browser.

Following is a description of all views available in Glia. Some of these are *redirect views*, which don't return a web page to the browser but redirect it to a different URL.

- **index** Frontpage at (http://rktik.com/)

Personas

- **create_persona** Form for adding a new persona
- **notebook** Private area for storing notes and reposting thoughts for oneself
- **notifications** Listing of notifications for active persona and email preferences for logged in user account
- **persona** Basic information about a single persona and listing of all movements they are a member of. This view includes a chat widget for private conversations with other users.
- **persona_blog** Personal blog of a persona

Thoughts

- **create_thought** Dedicated page for creating new thoughts. In contrast to the inline thought creator this view also allows entering long text attachments.
- **edit_thought** Similar interface to the *create_thought* view that allows removing and editing attachments and the thought's title.
- **delete_thought** Confirmation dialog for removing thoughts.
- **thought** View for a single thought that includes its context, attachments, comments and the thought's metadata.

Movements

- **movement** Redirect view that sends members to a movement's mindspace and non-members to the movement blog.
- **movement_blog** Main listing for a movement's blog that presents a reverse chronological, paginated view of thoughts in the blog. Also allows following the movement and becoming a member.
- **movement_mindspace** View for movement mindspace contents as well as movement chat and basic movement metadata.
- **invite_members** Form for obtaining invitation links for a movement and sending email invitations.
- **movement_list** List all movements registered on Rktik

User account related

- **activate_persona** Redirect view that activates a different persona registered to the logged in user account
- **signup** Form for creating a new user account
- **signup_validation** Redirect view for validating a user's email address
- **login** Login form for user accounts
- **logout** Redirect view that logs out the current user account

Helpers

- **help** Access help pages stored in *templates/help_*.html* files
- **tag** Listing of thoughts marked with a specific hashtag

Before request

These special views have the `before_request` decorator, which causes them to be executed every time a user visits a page.

- **account_notifications** Inserts a notification into the page if the logged in user has not validated their email address
- **mark_notifications_read** Marks all notifications as *read* which have a URL equal to the current page

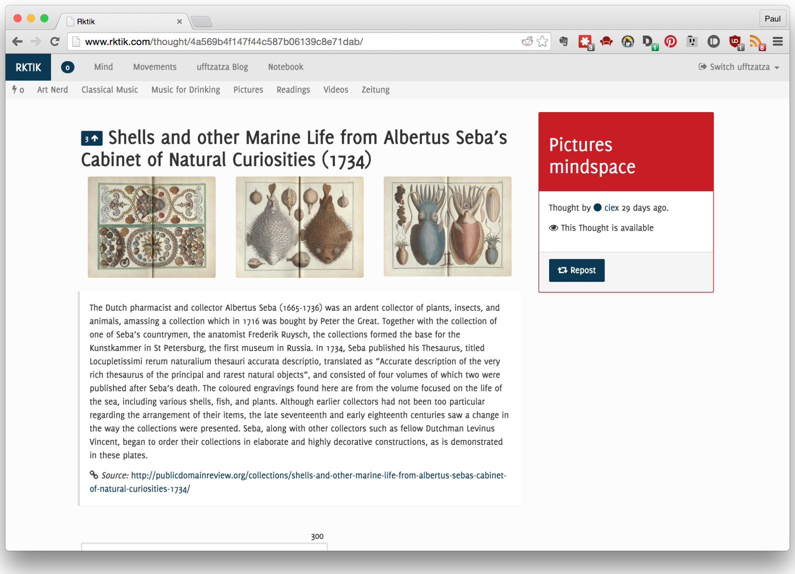
3.3.2 *HTML Templates*

Templates allow separation of content and layout in the application backend and thereby lead to more readable code. They consist of layout definitions written in HTML and additional markup that defines where content needs to be inserted. View functions compute all information necessary for a given web page and then pass this information as parameters to a template. Rktik uses the Jinja2 template engine[5] included with Flask for this purpose.

Jinja2 provides almost all required functionality with missing features provided through extensions. Rktik uses the *humanize* library[6] for converting date and time data into a human readable format.[7] Additionally, a number of custom filters are used in templates:

- Rktik stores all date and time information in the GMT timezone. This allows handling time information in the backend without considering time zones. A custom filter is used to convert these to European central time in the template.
- The *mentions* filter uses information from mention percepts (see Nucleus Models) to replace occurrences of the pattern `@persona_name` with a link to the respective persona's page
- The *gallery_col_width* filter is used to adapt the size of image attachments to their number. The largest format is used when only one image is attached. A successively smaller image size is used up to four image attachments.

5 Jinja2 2.8
6 Humanize 0.5
7 As an example, instead of displaying `2015-10-01T15:42:23.254966+00:00` as a thought's creation time, the relative form *two hours ago* is used



Shells and other Marine Life from Albertus Seba's Cabinet of Natural Curiosities (1734)

Thought by ciex 29 days ago.

This Thought is available

The Dutch pharmacist and collector Albertus Seba (1665-1736) was an ardent collector of plants, insects, and animals, amassing a collection which in 1716 was bought by Peter the Great. Together with the collection of one of Seba's countrymen, the anatomist Frederik Ruysch, the collections formed the base for the Kunstkammer in St Petersburg, the first museum in Russia. In 1734, Seba published his Thesaurus, titled Locupletissimi rerum naturalium thesauri accurata descriptio, translated as "Accurate description of the very rich thesaurus of the principal and rarest natural objects", and consisted of four volumes of which two were published after Seba's death. The coloured engravings found here are from the volume focused on the life of the sea, including various shells, fish, and plants. Although earlier collectors had not been too particular regarding the arrangement of their items, the late seventeenth and early eighteenth centuries saw a change in the way the collections were presented. Seba, along with other collectors such as fellow Dutchman Levinus Vincent, began to order their collections in elaborate and highly decorative constructions, as is demonstrated in these plates.

Source: http://publicdomainreview.org/collections/shells-and-other-marine-life-from-albertus-sebas-cabinet-of-natural-curiosities-1734/

- The *sort_hot* filter can be used to apply the hot ranking to lists of thoughts
- The *authorize* filter replaces thought contents with a placeholder if the thought is not visible to the active persona (see Serializable).

3.3.3 *Asynchronous UI*

Most of the content of Rktik is compiled on the server and then sent as a complete web page to the user's browser. When an interaction requires only part of the screen contents to be changed, site responsiveness is increased by using asynchronous communication with the server. This functionality is implemented using the *jQuery* Javascript library (jQuery website) for one-off asynchronous calls and the *websockets* browser technology for continous streams of information.

The websockets technology provides a socket between the browser and the Glia server which is used for relaying information during the time in which the browser window stays open. All thoughts created in a movement mindspace, which includes chat messages as well as all votes on these thoughts, are received and immediately relayed by the server to all browser windows that show part of the movement. Messages received on the client side are inserted into the chat widget. If the browser window shows an invidual thought's page, new comments are inserted at the appropriate place in the hierarchical comments view.

The same channel is used for sending reposts and receiving desktop notifications (see Notifications). Server side handlers for websockets are located in the `glia.web.events` module, while client side handlers are located in the static file `glia/static/js/main.js`.

Other asynchronous calls are handled using jQuery based javascript functions. This includes :

- Loading more chat contents.[8]
- Changing a movement's mission description.
- Changing a persona's username.
- Changing the amount of context[9] displayed above the thought title on individual thought pages.
- Following and unfollowing blogs.
- Toggling membership in a movement

The server side handlers for this functionality are located in the `glia.web.async` module, while the client side handlers are located in the `glia/static/js/main.js` script.

3.3.4 *Notifications*

Notifications are direct messages from the system to a user and inform them about reactions to their thoughts, as well as other relevant messages. They are relayed as desktop notifications and/or as email messages. Email messages provide the further advantage of being a way to contact users who are not visiting the site regularly.

Desktop notifications are displayed using the PNotify library (PNotify), which can insert notifications as HTML elements in web pages or

8 The chat window initially loads the most recent 50 messages, at the top of which a button triggers loading another 50

9 The context of a thought is that thought to which it is a reply. Applying this definition recursively gives a context depth.

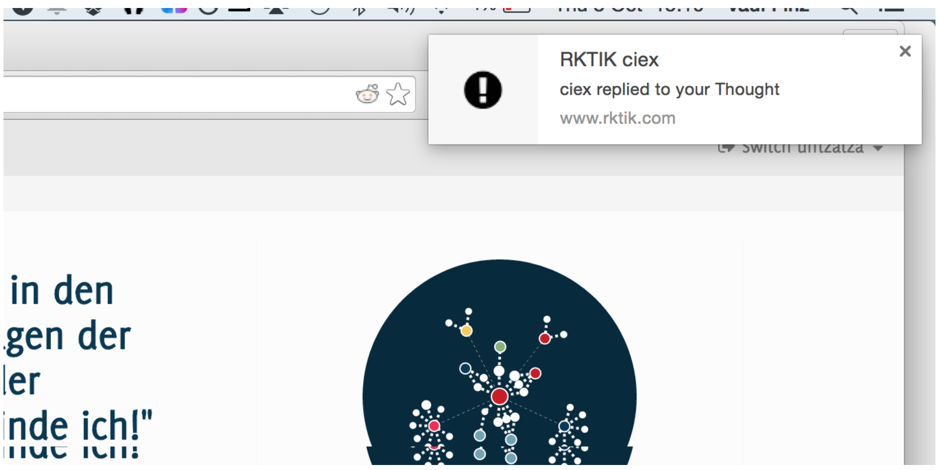

RKTIK ciex
ciex replied to your Thought
www.rktik.com

Figure 5: Desktop notification indicating a new reply on a user's thought

as operating system specific native UI elements outside the browser window, as specified in the W3C recommendation *Web Notifications* (see 12). When a user first visits the Rktik website, they are prompted to allow displaying web notifications. HTML based notifications are used if this request is denied. Desktop notifications are relayed to the browser using websockets. The javascript functions used for receiving and displaying notifications are located in the `glia/static/js/main.js` script.

Email notifications are delivered using the *SendGrid* email delivery service (SendGrid website). Using this service ensures that all users can receive email notifications reliably. While an email implementation integrated with the Rktik service would be technically feasible, this approach would not guarantee that messages pass spam filters of users' email providers. This functionality is implemented in the `glia.web.helpers` module.

The user may opt out of email delivery entirely or setup specific rules for the kind of emails they want to receive. These settings can be made in the notifications view which is linked from the notifications drop-down and from the footer section of all sent email notifications.

3.4 IMPROVING PERFORMANCE

User satisfaction is related to a web site's performance (5). As the complex page layouts and *hot* sorting used in Rktik require significant server resources, keeping performance at a satisfactory level is hard. As the development process of Rktik did not define performance as a primary objective (see Methodology), the necessary adjustments are even more difficult to make. Still, it was possible to increase performance at the end of the development process by 1) using memcache to reuse computed results and 2) optimizing database queries.

Caching

The memcache system is an in-memory key value store to hold computed results for fast access until they are overwritten, their expiry date is reached or they are deleted because available memory is not sufficient for new entries. Rktik uses the Flask-Cache library to access memcache (Flask-Cache 0.13.1 website). Caching is used if results 1) are changing slowly[10] , 2) are expected to be reusable in the near future and 3) can be reliably invalidated once they change.

Cached data is invalidated by processes that change its contents. See [Cached Information] for a list of cached functions and the processes that trigger their invalidation. Cache contents are automatically invalidated after an amount of time that ranges from minutes to days.

Database Query Optimization

The SQLAlchemy library hides the complexity of accessing data stored and linked across multiple tables. While this eases the development process significantly, it can lead to inefficient patterns of database usage. Specifically, the number of queries required to access data can be in a linear relation to the number of items retrieved. Often, such queries can be combined into a single query or a low number of queries by using *eager loading* techniques offered by SQLAlchemy (10). Here, a *JOIN* statement is issued to simultaneously load related data from the database. Rktik uses eager loading 1) ad hoc using the SQLAlchemy `joinedload`

10 As an example, the frontpage graph structure is cached for 1 hour per persona as the frontpage changes slowly and omissions are not considered critical.

option for specific queries and 2) in general by specifying relations to always load joined in model definitions located in Nucleus.

3.5 HOSTING AND DEPLOYMENT

Rktik ist running on servers provided by the Heroku platform-as-a-service (PAAS) ^[Heroku website). In contrast to traditional server environments, which need to be manually configured for the services to be deployed, a PAAS offers tools that automate many of these tasks. This includes deployment from a Git repository, automatic installation of dependencies, a web interface for installation and semi-automatic configuration of third-party services (e.g. email delivery, memcache, log analysis, etc.) and a mechanism for simple and fast scaling of an application's resources in the event of rising visitor traffic.

These capabilities allow a developer to focus on programming, instead of the time-consuming configuration and maintenance of a server environment. The downside of using Heroku is that their services come at a comparatively high price. This is mitigated somewhat as they are offering a free option for applications that require only little resources, as is the case for Rktik right now. However, if Rktik grows to a larger userbase, it might become neccesary to move to a different hosting environment that offers a better cost-benefit ratio.

Rktik is installed in two separate Heroku environments for *testing* and *production* use. The development process consists of testing a new feature on a local development machine, testing it in the testing environment and only then deploying it to the production environment if no bugs are found (see Methodology). The production environment has been accessible to the general public since 26th July 2015.

Deployment to these environments is automated using the scripts 'push_testing.py' and 'push_production.py' in the source code's root folder. These scripts execute all tasks necessary for deployment, which includes:

- Verifying that all changes to the Nucleus repository are commited in Git

- Pushing the Nucleus repository to Github
- Checking that all changes to the Glia repository are commited in Git
- Pushing the Glia repository to the appropriate environment on Heroku. This step triggers the Heroku environment to automatically install all required dependencies.[11]
- Running database migration scripts in the Heroku environment

The script for deployment to the production environment additionally merges all changes in the *development branch* into a new commit on the *master branch* of the Glia repository (see Methodology). Therefore, commits on the master branch represent a history of deployments to the production environment and can be seen as versions of the Glia application.

Rktik uses *environment variables* to determine which environment it is currently running in and to load an appropriate configuration. Configurations for the development, testing and production environment differ in the passwords, secrets and external services they specify as well as the internet hostname they setup for Rktik. Sensitive information, such as passwords, is not stored in the source code repository, but loaded either from access-controlled external files or from environment variables.

The development and testing configurations additionally increase the verbosity of log messages and provide interactive debugging in two ways: 1) a debugging console and interactive traceback embedded in Flask's response when server errors occur during a request and 2) the Flask DebugToolbar (11), which provides an interface for performance measurement, variable introspection and other information as an optional web overlay for successful requests.

All logging messages above the *debug* severity level are also forwarded to the Rktik engineering channel in the Slack web service,[12] which is not visible to the public. This allows monitoring of errors in Rktik from a mobile phone or any computer with an internet connection.

11 Dependencies are defined in the file `requirements.txt` in the Glia project root folder

12 The Slack web service provides chat rooms which can be integrated with external web services

4

EVALUATION AND DISCUSSION

Rktik is an evolution of the Souma prototype, introducing changes in some areas and taking a completely different approach in others. In this section I will reflect critically on how Rktik measures up to expectations set at the onset of its implementation and underline these findings with data from quantitative usage metrics.

4.1 PERSONAL EVALUATION

The goals of **completeness**, **feasibility** and **maintainability**, as defined in section Methodology, have been reached by the implementation described in this document. Development of the necessary features for *publishing* and *discussing* content using personas and movements have been built, and an appropriate environment for the operation of Rktik has been configured. While these initial goals are important, other aspects of Rktik have shown shortcomings in the course of development and testing. In this areas I have identified three aspects based on my own experiences and informal feedback received from acquaintances.

1. Some of the concepts used in Rktik, such as the distinction between a private mindspace and a public blog or the concept of movements, are hard to understand for new users. This problem should be approached by building a tutorial, which explains core mechanics and concepts. It can be presented after signup and should be directed both at new users with an account and potential, but unregistered users.

2. As described in the section Improving Performance, the site's speed is still too low to be satisfactory for many users. This can be improved by setting up periodically running background processes, which precompute database queries and store them in memcache. Site performance could also be improved significantly by moving Rktik to more powerful servers, which increases the costs of running the service.
3. Rktik features interesting content that is regularly updated. Users who have signed up for an account, but don't visit the site regularly, could benefit from this through a periodic email newsletter. Each edition of the newsletter may feature the most interesting content on Rktik as measured by votes received on thoughts.

4.2 USAGE METRICS

Usage metrics have been collected anonymously using the Amplitude and Google Analytics services. These are external services which receive usage data via short Javascript scripts embedded in every served page. As I consider this usage data *personal information*, I have collected it anonymously. While Google Analytics tracks information about page requests, Amplitude tracks specific, manually defined events that can occur after a page was fully loaded.

Google Analytics data is included since making rktik.com accessible to the public on July 26th 2015 until October 2nd 2015 for a total of 68 days. Amplitude data is included from August 8th 2015 until October 2nd 2015 for a total of 55 days. This difference exists because I only found out about Amplitude when measurement had already begun.

Google Analytics and Amplitude collect extensive amounts of information. As a complete analysis of the data would go beyond the scope of this thesis, I have focussed on four kinds of measurement: 1) The total number of usage sessions, 2) the number of users per week, 3) the number of users who have used specific features in a given week and 4) the number of thoughts and votes created. I chose these metrics because they reflect both the total amount of usage as well as the relative popularity of specific features.

For the purpose of this evaluation, a *user* is defined as a web browser used to access Rktik. In contrast to the definition given in Terminology, this does not discriminate between users with an Rktik user account and anonymous users. While it is possible that a person uses more than one web browser to access Rktik, this is not differentiated as it is not technically feasible to identify this behavior. Usage assessment using Google Analytics and Amplitude relies on Javascript scripts embedded in the site. If a user has blocked these scripts or disabled Javascript altogether in their browser, their usage is not recorded.

4.2.1 *Usage sessions*

The number of sessions was assessed using Google Analytics. A session is defined as a group of interactions by a user that takes place within 30 minutes or until midnight (3). A total of 825 sessions were started, most of which took place in the first weeks of operation.

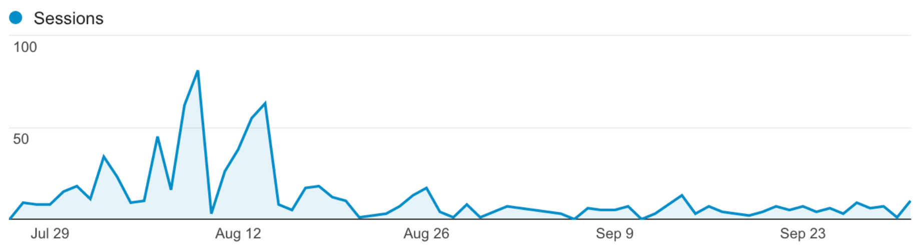


Figure 6: Number of sessions

4.2.2 *Users per week*

The number of users per week was assessed using Amplitude. The count rises from two users in the first week of measurement to seven users in the third and fourth week, takes a plunge to three users and then rises from week to week up to a maximum of 17 users in the 9th week of operation.

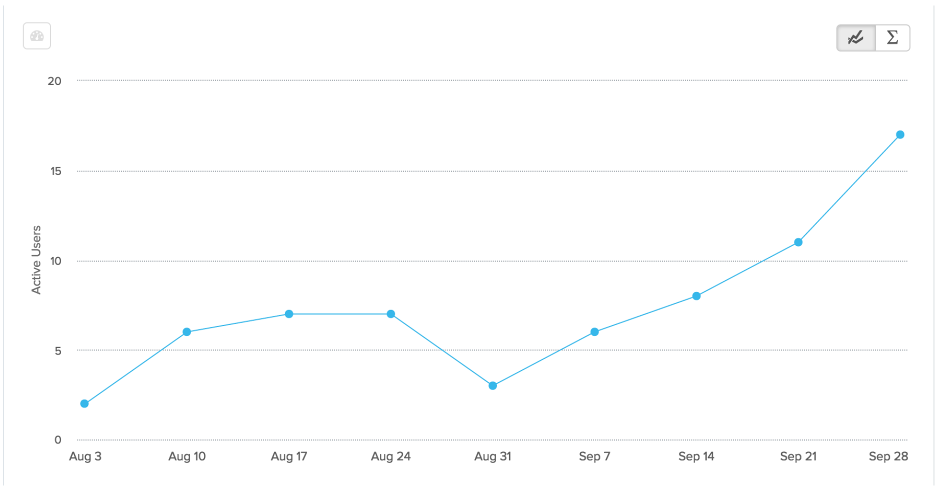


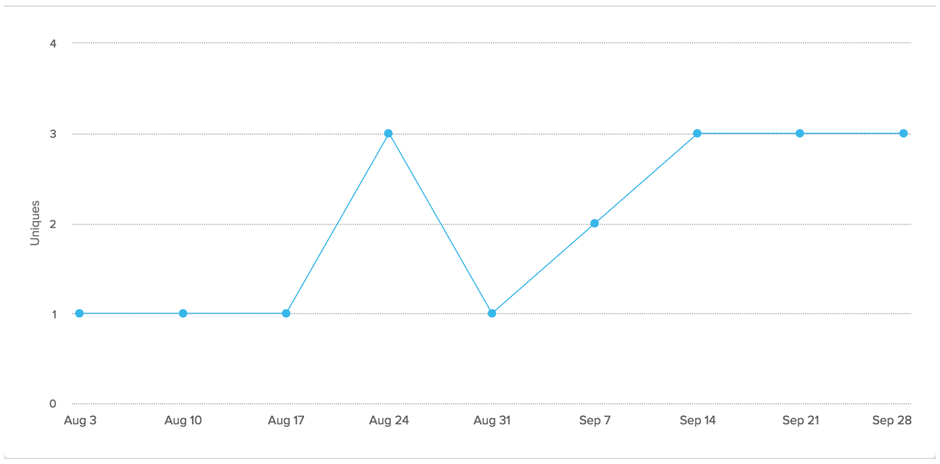
Figure 7: Number of weekly users

4.2.3 *Usage of features*

The number of users who have used specific features at least once in a given week was assessed using Amplitude. A count was recorded by firing an Amplitude callback when a user triggers the user interface control associated with the measured action. In the following I list the assessed metrics, events used to measure them and describe the resulting measurement data.

4.2.3.1 *Creating a thought*

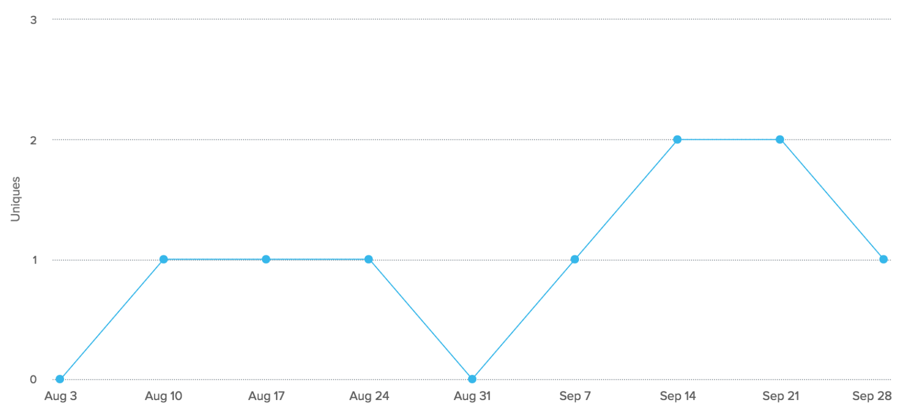
Evaluated: 1) submitting the inline form for creating a thought 2) submitting the form on the dedicated *create thought* view



The number of users who create thoughts in a given week varies between one and three over the duration of measurement. The maximum number of three users was first reached in the fourth week of measurement and in the last three weeks of measurement there were never less than three users creating thoughts. While only a small amount of total users register an account, only a small amount of registered users actually create content on the site.

4.2.3.2 *Editing a thought*

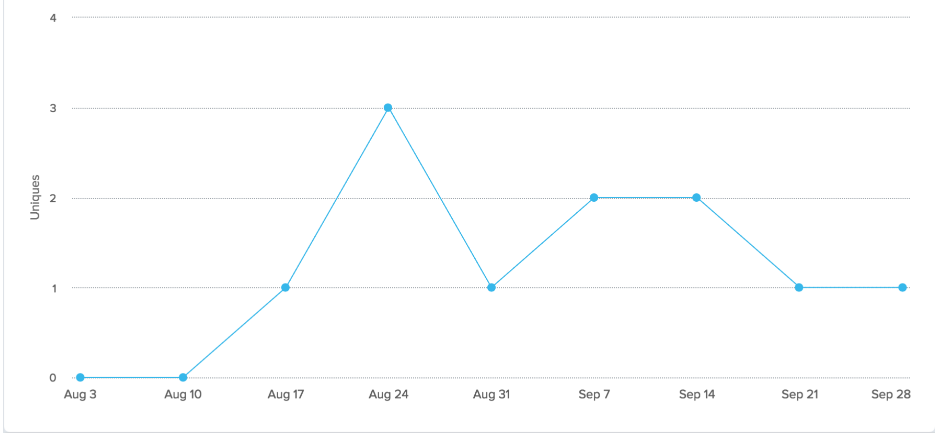
Evaluated: Submitting the form for editing a thought in the *edit thought* view.



The number of users who edited thoughts in a given week varies between zero and two users. Unexpectedly, this number is not much lower in a given week than the number of users who created thoughts which means that most users who create thoughts also edit them.

4.2.3.3 *Clicking the graph visualization*

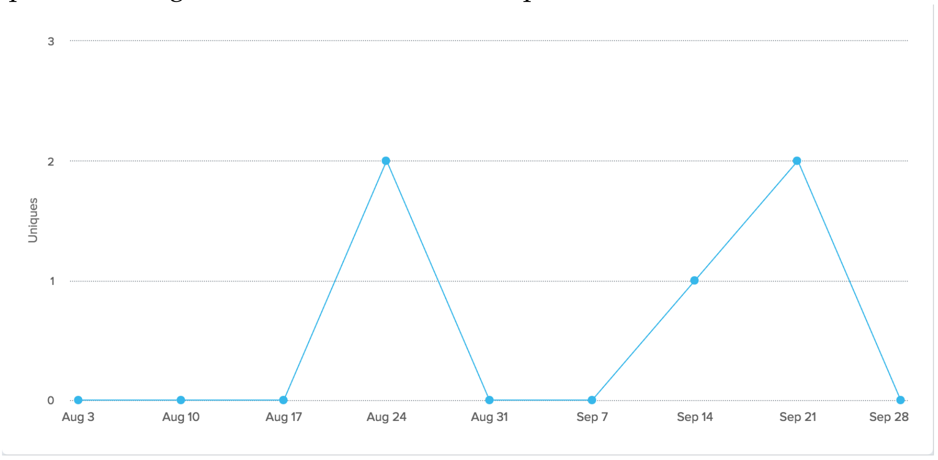
Evaluated: Completed mouse click in any part of the frontpage graph visualization (see Frontpage Graph Visualization).



The number of users who clicked on the frontpage graph visualization in a given week varies between a minimum of zero users in the first two weeks and a maximum of three users in the fourth week of measurement. As the number of of clicks does not return to the maximum of three users and rests at one user per week in the last two weeks of measurement, it may be assumed that users try this feature only once and don't return to it later.

4.2.3.4 *Follow or unfollow a blog*

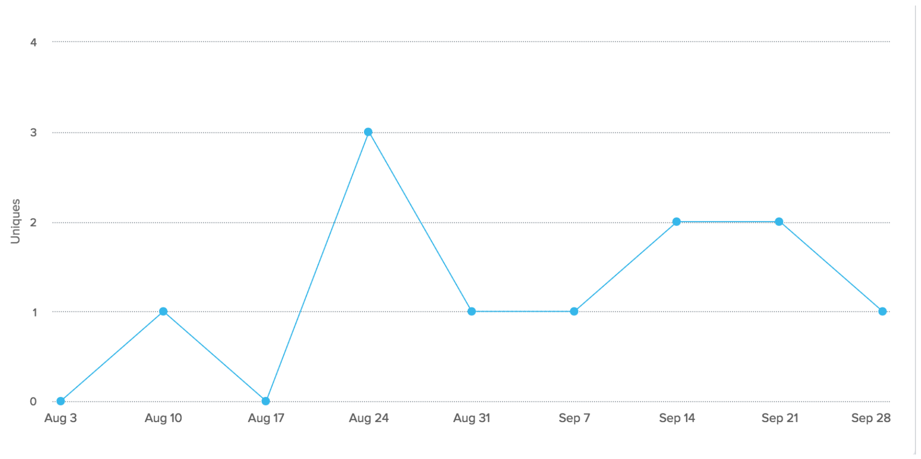
Evaluated: Click on the *follow* or *unfollow* buttons located in movement or personal blogs and in movement mindspaces.



The number of users who follow or unfollow blogs lies between zero and two users over the course of measurement. This number is unexpectedly low in relation to the total number of active users, especially towards the end of the measurement. This may indicate that users don't understand the purpose of this feature well enough. Educating users about the possibility of controlling the contents of their frontpage by following or unfollowing blogs may lead to a better user experience for them.

4.2.3.5 *Toggle membership in a movement*

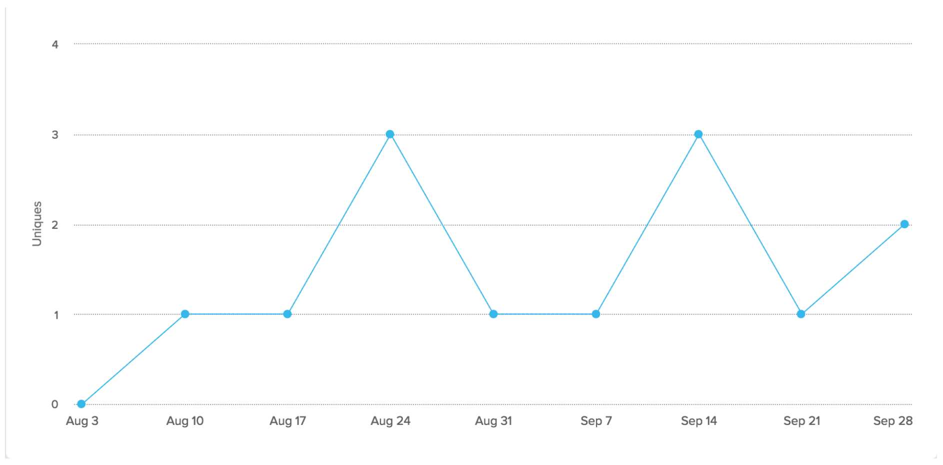
Evaluated: Click on the *Join movement* or *Leave movement* buttons located in movement blogs and movement mindspaces.



The number of users who toggle their membership in movements is 1-2 in the first three weeks, reaches a maximum of three users in the fourth week and varies between one and two users for the remainder of the measurement. Similar to the number of users who follow or unfollow blogs, this number is unexpectedly low and suggests that users don't understand the benefits of controlling their membership in movements well enough.

4.2.3.6 *Accessing the notebook view*

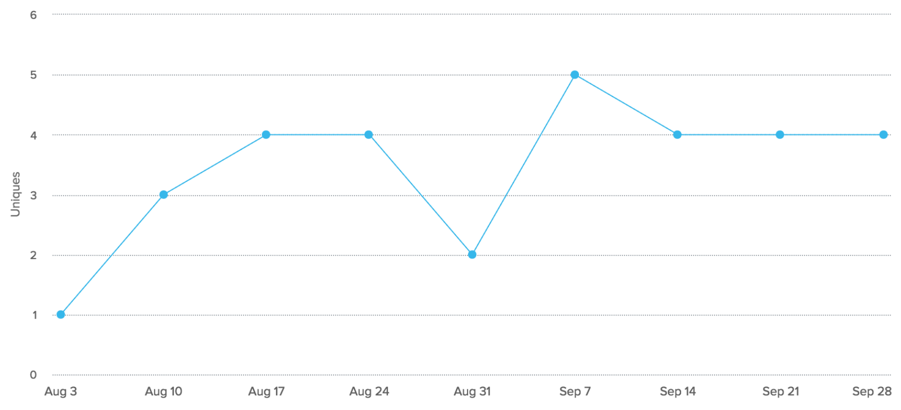
Evaluated: Loading the notebook view



The number of users who view their notebook is zero in the first week and varies between one and three users for the remainder of the measurement. The feature was not used in the first week because it was only activated in the second week of measurement. Engagement with this feature may be higher if it was more integrated with other features of the site. Right now, the flow of data in and out of the notebook is achieved via the *repost* functionality. This mechanism may be poorly understood by users which could be helped by a more comprehensive tutorial for new users.

4.2.3.7 *Toggling a vote*

Evaluated: Completed mouse click on a vote button



The number of users who cast votes rises from one in the first week to four in the third week and remains at this level except for the 5th and 6th weeks, where two and five users engaged with the feature respectively. This feature is the most popular in this comparison, which confirms its nature as a low-barrier way of engaging with content in Rktik.

4.3 MOVEMENT AGENCY

Rktik communicates to its users that movements have agency. This is established by attributing actions taken by its members to the movement itself, once they are confirmed by a certain number of other members. As described in Movements, this process 1) guards members' privacy and 2) may establish stronger cohesion between movement members.

While this concept may be applicable to a wide range of actions, Rktik's current implementation only supports attributing authorship of a thought to a movement (see Promoting Content). In this section I will present three options for other actions that may be attributed to a movement's agency through building consensus among its members.

Planning and voting of any such collective actions may be conducted using a novel interface control which allows members to 1) identify the proposed action, 2) propose changes and 3) vote on its implementation.

This control should be designed for the control of all action types to simplify the process for inexperienced users.

Events

A movement member may propose an event in the movement mindspace. An event consists of a title, location, date, time and privacy setting (visible only for members or for all users) and can be implemented as an attachment type. Other members may then discuss the events and propose changes to be made by the author in order to gain their consensus, and thus the necessary votes for making the event official. Finally, members may vote on the event. If this show that consensus is reached, the movement would display the event prominently in the movement mindspace or movement blog, depending on the event's privacy settings. The final event posting would be attributed to the movement and not its original author. This mechanic would allow movement members to organize collective action outside of Rktik's online context.

Public chat

A movement's (public) blog may contain a chat module, which nonmembers can use to converse with the movement as a whole. Messages entered in this chat would be displayed in the movement mindspace for members to see. Members can then propose answers and other members can confirm these by voting on them. The non-member who posted the original message into the movement chat would then be shown the answer attributed to the movement as a whole.

Creating thoughts in other parts of the site

Members may propose a thought to be posted with attribution to the movement in another part of Rktik. This could be a personal message to a persona, a reply to any thought not in the movement itself or as a message in the public chat of another movement (as described above). Changes to the proposed thought may be discussed by members beforehand as described for *events* and *public chat* .

4.4 EXTERNAL CLIENTS

Rktik is the successor to Souma, which is a prototype for a decentral social network as described in Problem: Privacy and Identity. While Rktik is not built as a decentral system, its design allows a future extension with such functionality. Such an extension would entail two parts: 1) a publicly accessible API for Rktik's Nucleus backend and 2) a local client application that connects to this API. Using a local client makes a new usage pattern possible, in which a private movement's internal data is only stored in encrypted form on the server. Encrypted information about members' activity would be distributed among them via the API. Information would only be made available in unencrypted form on the rktik.com website once members have decided on a collective action. In this section I will outline the components necessary to build this extension based on the design implemented by Souma (see 1).

API for Rktik's Nucleus module

An API is an interface between two separate pieces of software. JSON and HTTP REST APIs have emerged as de facto standards in web infrastructure (CITE). The API will serialise ORM models into JSON representations, which are encrypted, signed and wrapped in a second layer JSON representation for transfer.

The API will be designed primarily for a client that is installed on a user's computer, but it may also be used for third party services built on data published by Rktik. Usage scenarios include clients for mobile devices or third-party websites that interact with Rktik.

Local client application

An external client uses the aforementioned API to exchange data with Rktik's server-based Nucleus backend and provides its own user interface on the user's local machine. The implementation of this user interface may reuse most of the code from Rktik's Glia web server to locally serve a version of the site via a loopback connection.[1] Differences in implementation would be expected mostly from 1) code to access the API and 2) performance optimizations for personal computers.

1 TODO: Footnote about loopback connections

While such an implementation of an external client would not technically constitute peer to peer data transfer, as data is still transferred using a server, it would bring all of its advantages with it: End-to-end-encryption of data ensures that the contained *information* is only available at the endpoints of communications. At the same time, the server-based infrastructure would 1) ensure that data is available, even when no local machines are online and 2) increase synchronization speeds significantly by profiting off the superior connection speeds available in data centers.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Ahrend, V., Berov, L., Eilbacher, G., et al. From a humanistic reconstruction of social networks to Souma. In L. Berov, A. Jamneshan and L. Wjatscheslaw, eds., *From a humanistic reconstruction of social networks to souma*. Not published, Osnabrück, 2015, 95–108.

[2] Facebook. Facebook Company Info. http://newsroom.fb.com/company-info/.

[3] Google. How a Session is defined in Analytics. https://support.google.com/analytics/answer/2731565?hl=en.

[4] Gruber, J. Markdown. 2004. http://daringfireball.net/projects/markdown/.

[5] NIELSEN, J. User Satisfaction vs. Performance Metrics. 2012. http://www.nngroup.com/articles/satisfaction-vs-performance-metrics/.

[6] Radicati, S. and Hoang, Q. Email statistics report, 2011-2015. *Retrieved May 44*, 0 (2011), 0–3.

[7] Reddit. About reddit. https://www.reddit.com/about/.

[8] Reddit.com. Reddit FAQ: Is it okay to create multiple accounts? https://www.reddit.com/wiki/faq/#wiki/_is/_it/_okay/_to/_create/_multiple/_accounts.3F.

[9] Salihefendic, A. How Hacker News ranking algorithm works. 2010. http://amix.dk/blog/post/19574.

[10] SQLAlchemy Autors. Relationship Loading Techniques. http://docs.sqlalchemy.org/en/rel/_0/_8/orm/loading.html.

[11] Tellingen, M. van and Individual Contributors. Flask DebugToolbar. 2015. https://flask-debugtoolbar.readthedocs.org/en/latest/.

[12] W3C. Web Notifications (W3C Proposed Recommendation). http://www.w3.org/TR/2015/PR-notifications-20150910/.

APPENDIX

.1 api specification

API Specification documents

.2 rights management

This table show which users/personas are allowed to make changes to specific models.

Rights Management for Mindsets

Mindset | Creating | Updating | Deleting

Personal Mindspace — Persona — Personal Blog — Persona — Movement Mindspace Members — Author, Admin — Movement Blog Automatic* — Admin — Dialogue Members — Author —

```
* see section ___ on auto-promotion from movement mindspaces to blogs
```

.3 cached information

This section gives an overview of values cached using memcache as described in section [Improving Performance].

Methods: * Persona.attention * Persona.conversation_list (invalidated by Thought.create_from_input) * Persona.frontpage_sources (invalidated by Persona.toggle_following, Persona.toggle_movement_membership) * Persona.movements (invalidated by Persona.toggle_movement_membership) * Persona.repost_mindsets (invalidated by Persona.toggle_movement_membership) * Persona.suggested_movements

- Movement.attention
- Movement.member_count (invalidated by Persona.toggle_movement_membership)
- Movement.mindspace_top_thought (invalidated by Thought.toggle_upvote)
- Movement.top_movements
- Thought.top_thought (invalidated by Thought.create_from_input)
- Thought.upvote_count (invalidated by Thought.toggle_upvote)
- Thought.iframe_url

Additional: * Recent thoughts helper Nucleus.helpers.recent_thoughts (invalidated by Thought.create_from_input) * “Percept” template macro * Frontpage graph visualization * Async chat view

.4 USAGE METRICS

.4.1 *Amplitude*

.4.2 *Google Analytics*

.5 SOURCE CODE